

7장. 데이터베이스 설계와 E-R 모델

이 동 호

데이터베이스 연구실
소프트웨어학부

목차

- 7.1 설계 과정의 개요
- 7.2 개체 관계 모델링
- 7.3 제약 조건
- 7.4 개체 집합에서의 중복된 속성 제거
- 7.5 개체-관계 다이어그램(E-R 다이어그램)
- 7.6 릴레이션 스키마로의 축소
- 7.7 개체-관계 설계에서 대한 논점들

7.1 설계 과정의 개요

- 설계 단계
 - 사용자 요구사항 명세
 - 개념 설계
 - 조직체에 대한 상세한 개관 제공
 - 개념 설계를 표현하기 위해 개체-관계 모델(**entity-relationship model**) 사용
 - 개념 설계의 결과물은 스키마에 대한 그래픽적 표현, 즉 E-R 다이어그램
 - 기능적 요구사항 명세
 - 데이터에 대해 필요한 연산(혹은 트랜잭션) 명세
 - 논리 설계
 - 개념적 스키마를 실제 데이터 모델로 변환 (즉, 스키마 정의)
 - 물리 설계
 - 파일 구성, 색인 사용등 물리적 특징을 명세

7.2 개체-관계 모델

- 데이터베이스는 아래 두 요소로 설계됨
 - 개체들의 집합
 - 개체들간의 관계
- 개체(**entity**)
 - 다른 객체와 구분될 수 있는 객체
 - 예) 사람, 회사, 사건, 설비 등
 - 객체는 속성을 갖는다.
 - 예) 사람은 이름과 주소라는 속성을 갖을 수 있음
- 개체 집합(**entity set**)
 - 동일한 속성을 공유하는 개체들의 집합
 - 예) 모든 사람들, 회사들, 나무들 집합 등

개체 집합(Cont.)

instructor_ID instructor_name

76766	Crick
45565	Katz
10101	Srinivasan
98345	Kim
76543	Singh
22222	Einstein

instructor

student-ID student_name

98988	Tanaka
12345	Shankar
00128	Zhang
76543	Brown
76653	Aoi
23121	Chavez
44553	Peltier

student

관계 집합

- 관계(**relationship**)는 여러 개체들 사이의 연결

예제:

44553 (Peltier)	<u>advisor</u>	22222 (Einstein)
student 개체	관계	instructor 개체

- 관계 집합(**relationship set**)
 - 2개 이상의 개체들 간의 만들어질 수 있는 관계들의 집합

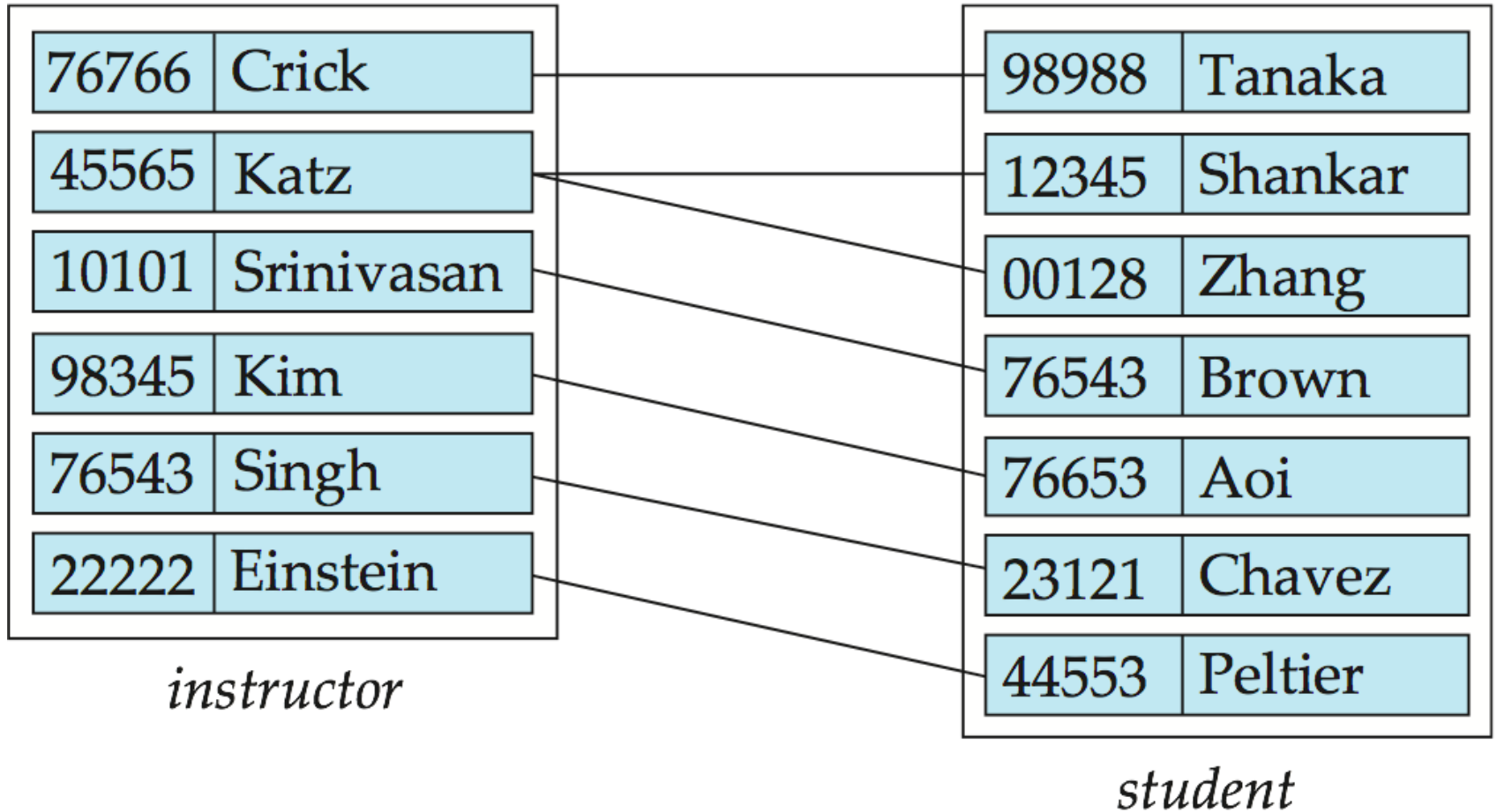
$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

여기서, $(e_1, e_2, \dots, e_n)$ 는 관계 집합

– 예제:

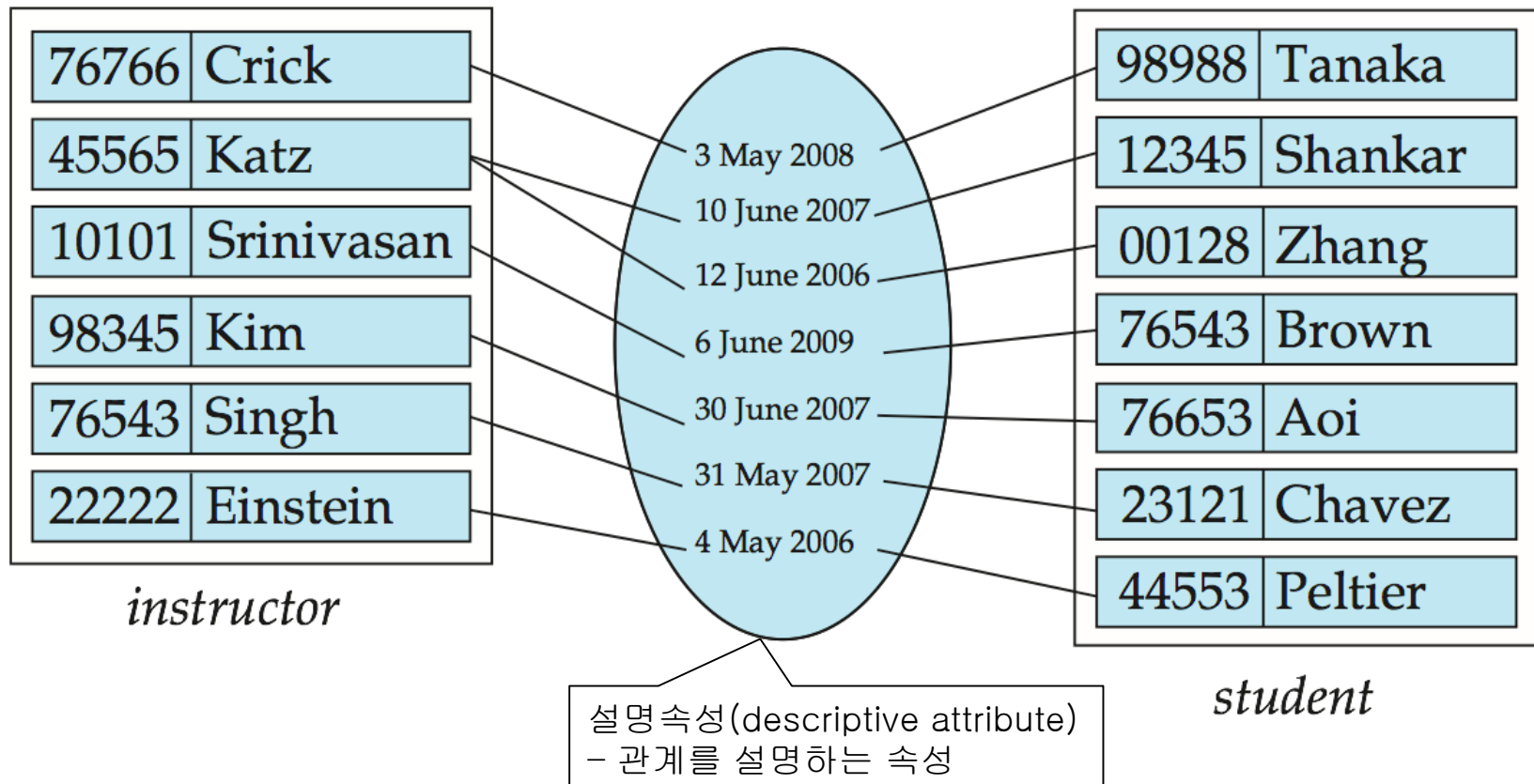
$$(44553, 22222) \in \text{advisor}$$

관계 집합(Cont.)



관계 집합(Cont.)

- 관계 집합에도 속성이 있을 수 있음
- 예를 들어, advisor관계 집합은 date라는 속성을 갖을 수 있는데 이것은 student와 instructor가 advisor 관계가 성립된 날짜임



관계 집합의 차수(degree)

- 이진관계(binary relationship)

- 두개의 개체가 연관됨 (차수 2).
- 데이터베이스에서는 대부분의 관계가 2진 관계임

- 두 개 이상의 개체과 연관되는 관계는 드물다. 간혹, 2진 관계 이상이 있을 수도 있음

- ▶ 예제: 3진 관계(*proj_guide*) - *instructor*, *student*, 그리고 *project* 간의 관계
- ▶ *students* 가 *instructor*의 지도를 받으면서 *projects*를 수행할 경우

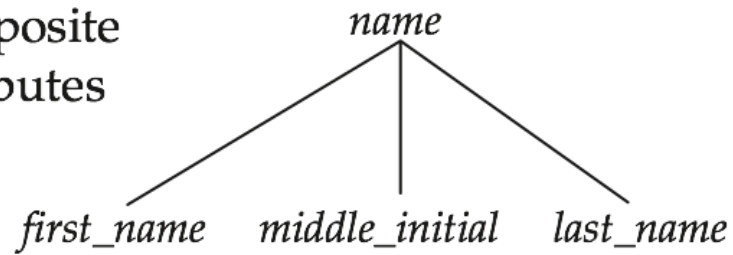
→ 모든 3진 이상의 관계는 2진 관계로 표현할 수 있음 (추후 살펴봄)

속성

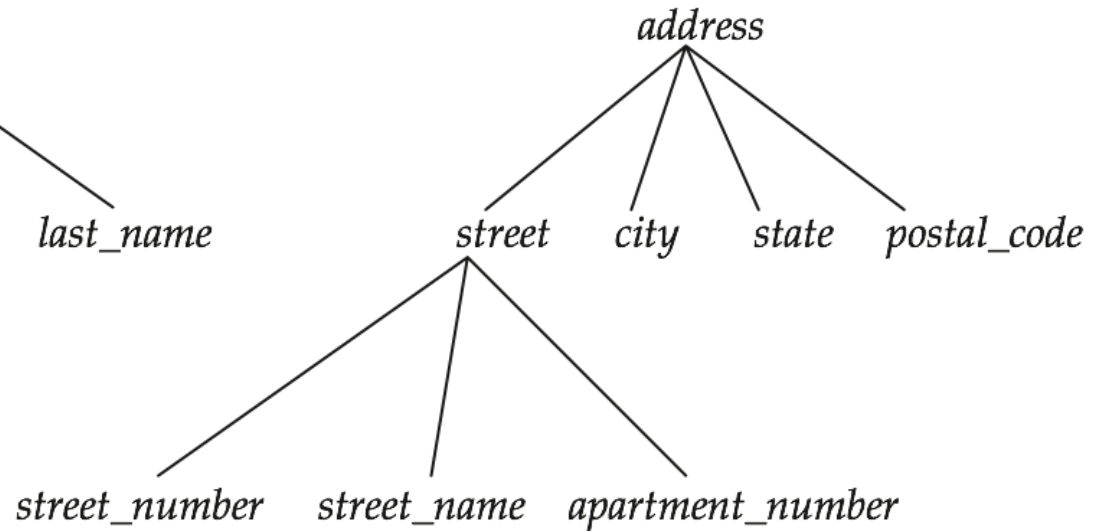
- 객체는 속성들의 집합으로 표현
 - 예:
instructor = (ID, name, street, city, salary)
course = (course_id, title, credits)
- 속성의 종류
 - 단순, 복합 속성
 - 복합 속성의 예: name (first, middle, last name으로 구성)
 - 단일-값, 다중-값 속성
 - 다중-값 속성의 예: *phone_numbers* (집, 사무실, 핸드폰)
 - 유도된 속성
 - 다른 속성으로부터 유도될 수 있는 속성
 - 예: *date_of_birth*로부터 *age*를 추론 가능

복합 속성

composite
attributes



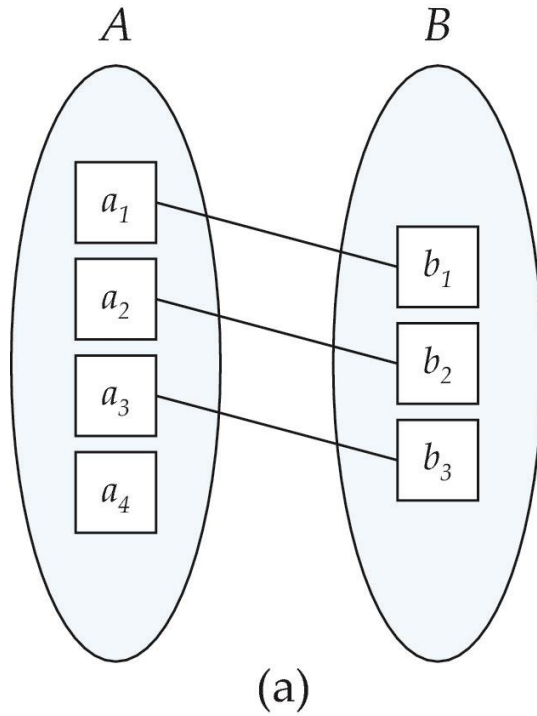
component
attributes



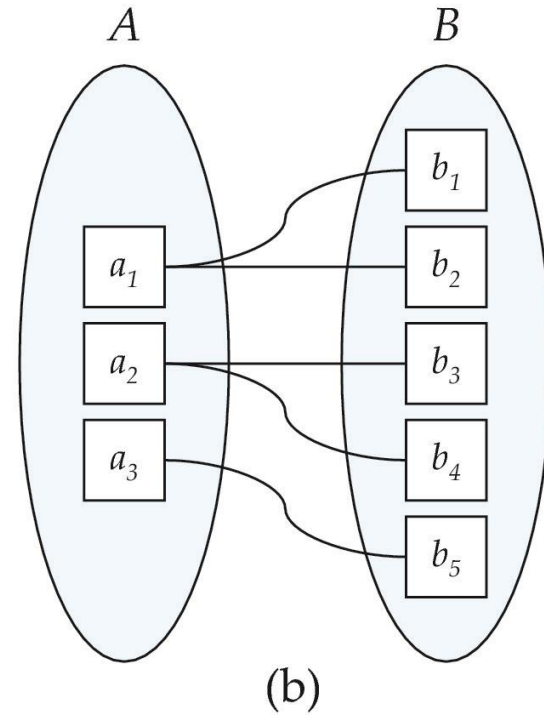
7.3 제약조건 – 대응수(Mapping Cardinalities)

- 관계 집합을 통해서 다른 객체와 연관될 수 있는 객체의 수
- 이진 관계에서 대응수는 다음의 4가지 경우가 있음
 - 일대일(One to one)
 - 일대다(One to many)
 - 다대일(Many to one)
 - 다대다(Many to many)

대응수(Cont.)



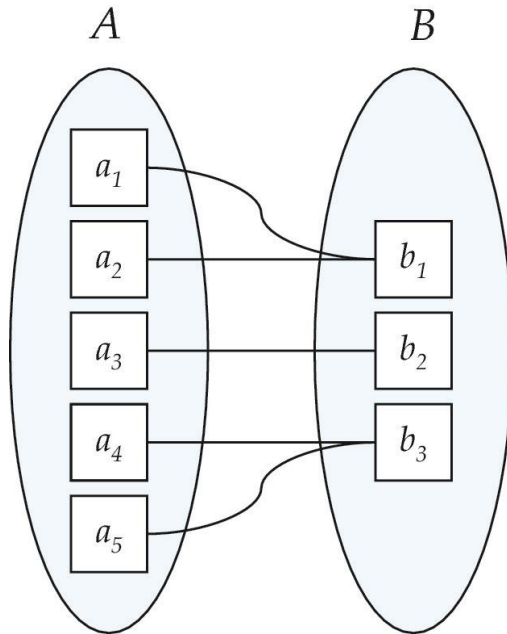
One to one



One to many

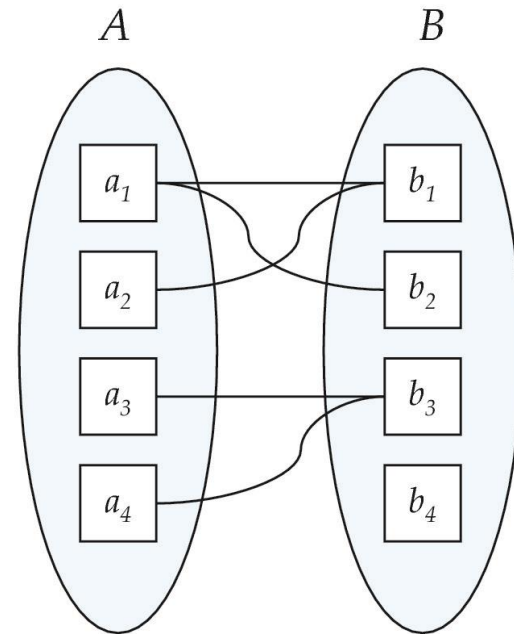
참고: A와 B에 있는 어떤 원소는 서로 대응되는 원소가 없을 수도 있음

대응수(Cont.)



(a)

Many to
one



(b)

Many to many

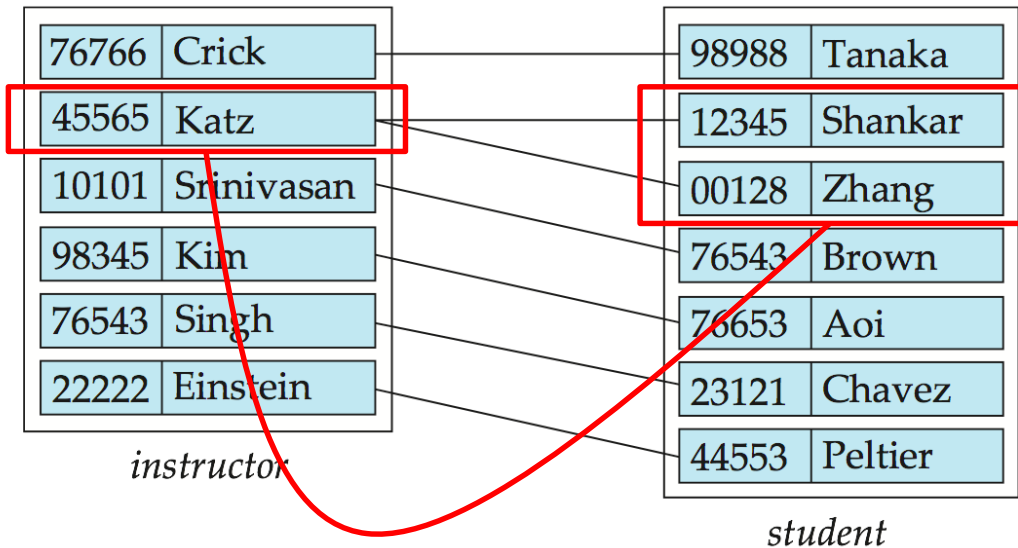
참고: A와 B에 있는 어떤 원소는 서로 대응되는 원소가 없을 수도 있음

개체 집합을 위한 키

- 수퍼키
 - 개체 집합에서 각 개체를 유일하게 구분해 낼 수 있는 하나 이상의 속성들의 집합
- 후보키
 - 더 이상 쪼갤 수 없는 가장 적은 개수의 속성들로 이루어진 수퍼키
- 주키
 - 여러 후보키 중에 데이터베이스 설계자가 선택한 하나의 후보키

관계 집합을 위한 키

- 관계 집합의 수퍼키는 관계에 참여하는 개체들의 주키의 조합으로 구성됨
 - (s_id, i_id) : *advisor* 관계 집합의 수퍼키
- 관계 집합의 후보키(혹은 주키)는 대응수에 따라 달라짐
 - *student-to-instructor* 관계집합
 - 다대일, 주키 : (s_id)
 - 일대다, 주키: (i_id)
 - 일대일, 주키: (s_id) or (i_id)
 - 다대다, 주키 : (s_id, i_id)
- 여러 개의 후보키 중에서 하나의 주키를 선정할 경우 관계 집합의 의미를 잘 고려할 필요 있음
 - 왜 그럴까? → 실제 관계 relation을 잘 생각해 보면 당연!



일대다

instructor-to-student

i_id	s_id
76766	98988
45565	12345
45565	00128
10101	76543
.....	

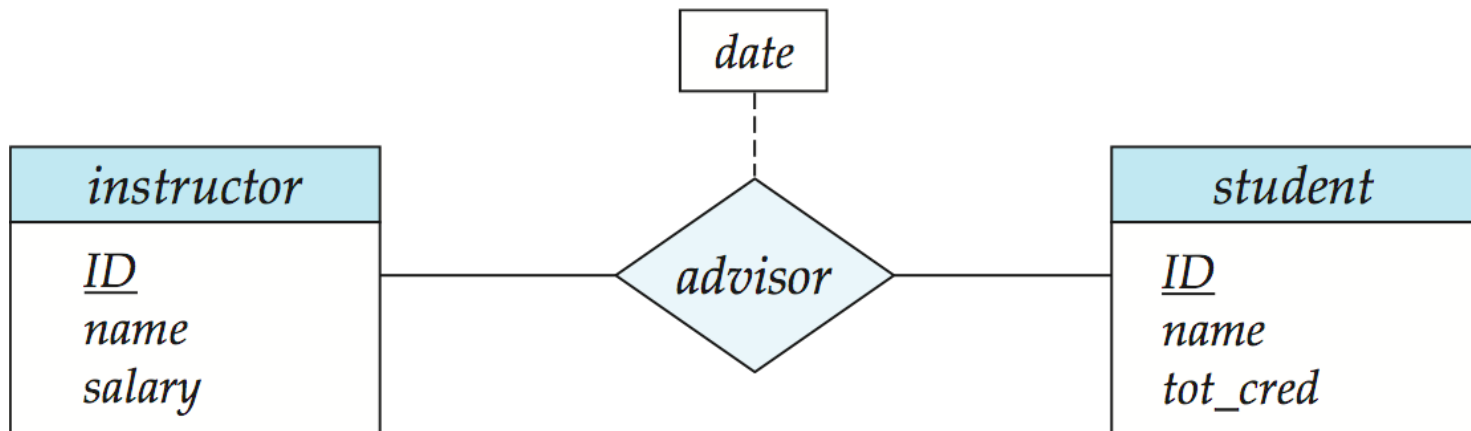
결국, many측 참여 개체의 Primary key가 관계 릴레이션의 Primary key로 사용됨

7.4 중복된 속성 제거

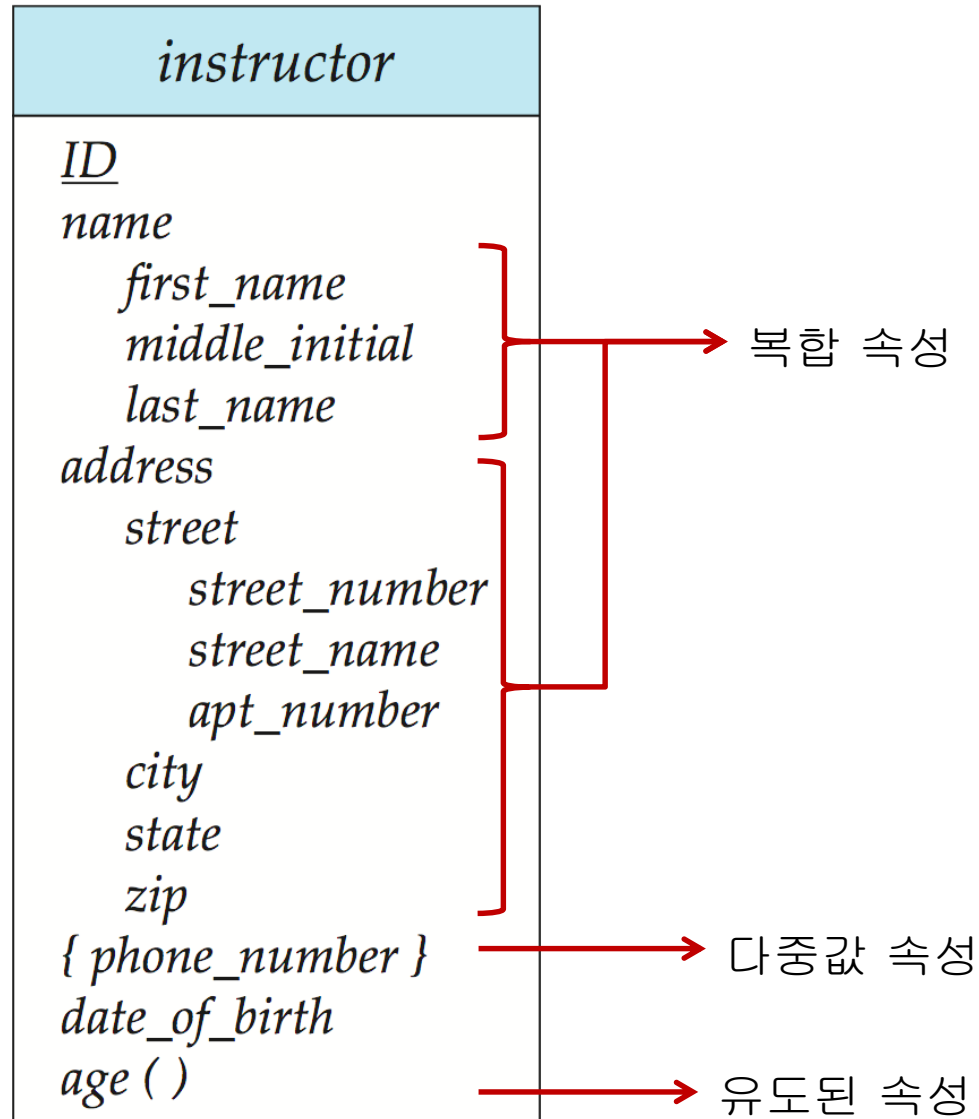
- 가정: 2개의 개체 집합, 1개의 관계 집합
 - 객체 집합: ***instructor***(주키: *instructor_ID*, 속성으로 *instructor* 가 소속된 *dept_name* 포함), ***department***(주키: *dept_name*)
 - 관계 집합: ***inst_dept***(*instructor* 와 *department* 을 연결)
- inst_dept* 관계에 *instructor_ID* 와/ *dept_name* 이 속성으로 존재하기 때문에 *instructor* 개체에 포함된 *dept_name* 속성은 중복됨
 - 따라서 *instructor* 개체에서 *dept_name* 속성을 제거할 수 있음
 - 참고: *instructor* 개체를 테이블로 만들 때, 필요에 따라서 *dept_name* 속성이 들어갈 수도 있음

7.5 개체-관계(E-R) 다이어그램

- 기본 구조
 - 사각형 : 객체 집합
 - 다이아몬드 : 관계 집합
 - 속성 : 사각형(개체 집합) 안에 표시
 - 밑줄 : 주키 속성 표시
 - 선 : 개체 집합과 관계 집합을 연결
 - 점선 : 관계 집합의 속성을 표시
 - 이중선 : 개체가 관계 집합에 전체 참가를 의미함
 - 이중 다이아몬드 : 약성 개체 집합에 연결된 식별 관계 집합을 표시

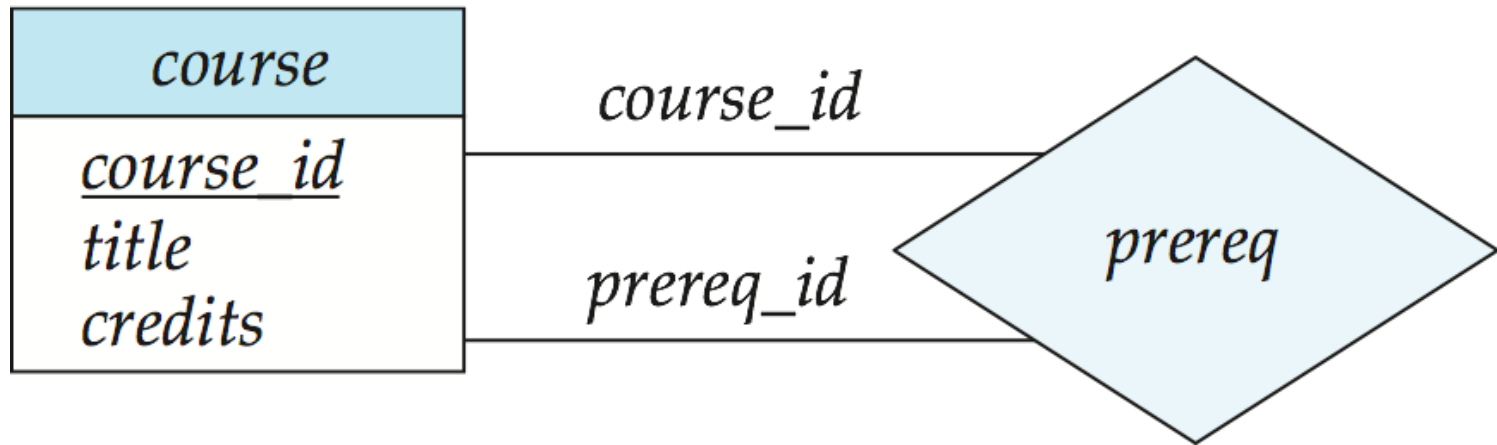


복합 속성



역할

- 관계에 참여하는 개체 집합이 반드시 다를 필요는 없음
 - E-R 다이어그램에서 사격형과 다이아몬드를 연결하는 선위에 레이블을 붙여 역할을 표시함
- 레이블 “*course_id*”와 “*prereq_id*” 은 역할을 의미



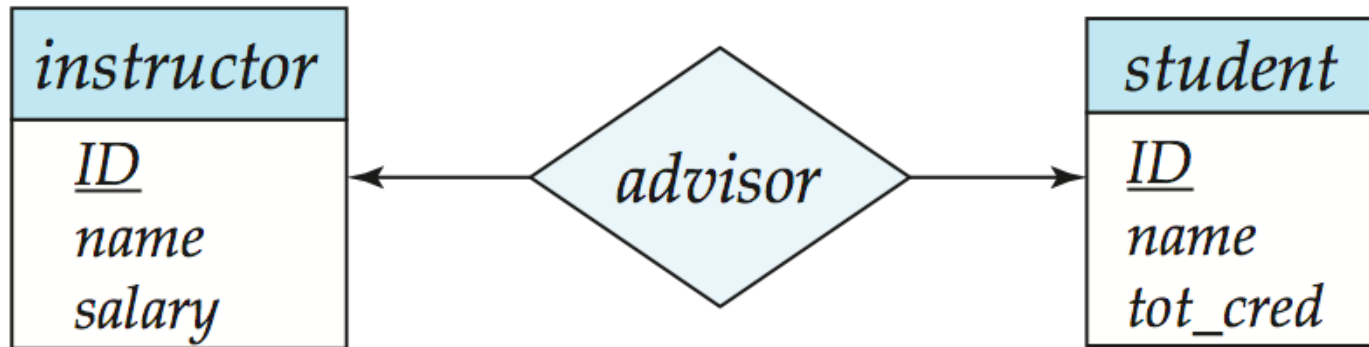
선위의 label → role(역할)을 표시

대응수 제약조건

- E-R 다이어그램에서,
 - 화살표 ($\rightarrow$) : 대응수 “일”을 의미
 - 일반선 (—) : 대응수 “다”를 의미

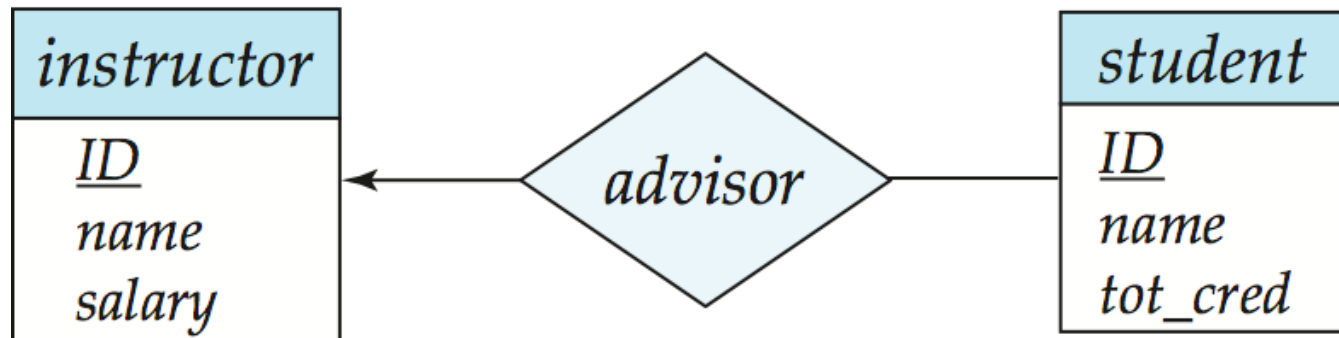
일대일 관계

- 한 명의 instructor는 advisor관계를 통해 많아야(at most) 한 명의 student와 관련
- 한 명의 student는 advisor관계를 통해 많아야 한 명의 instructor와 관련



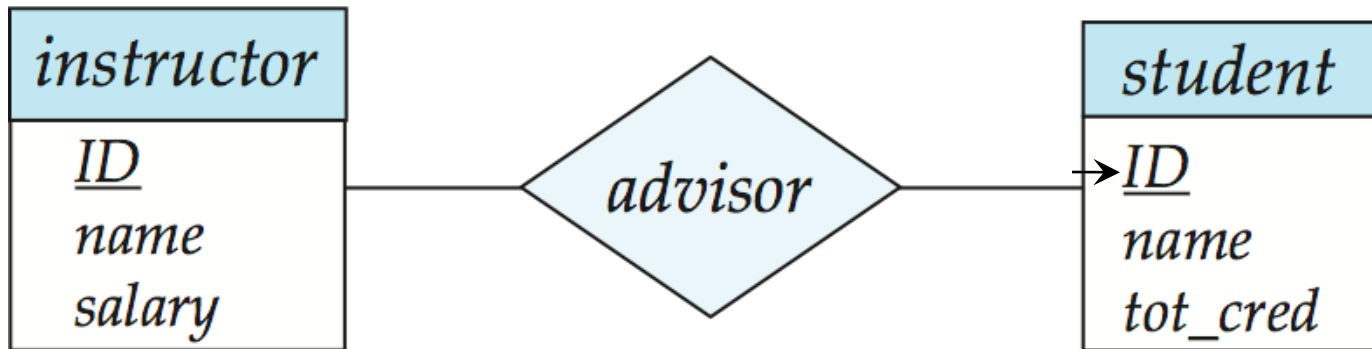
일대다 관계

- 한 명의 instructor는 advisor관계를 통해 여러 명(0명 포함)의 student들과 관련
- 한 명의 student는 advisor관계를 통해 많아야 한 명의 instructor와 관련



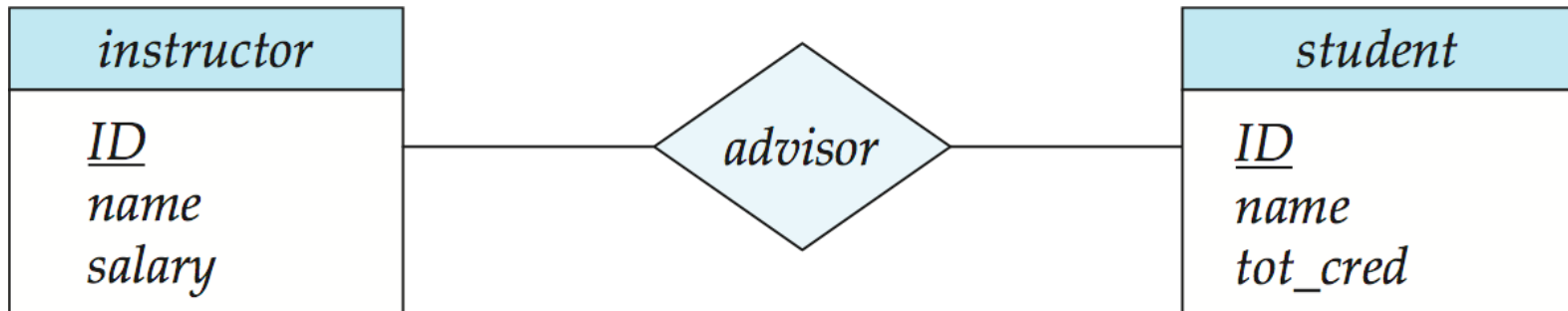
다대일 관계

- 한 명의 instructor는 advisor관계를 통해 많아야 한 명의 학생과 관련
- 한 명의 student는 advisor관계를 통해 여러 명의 instructor들과 관련



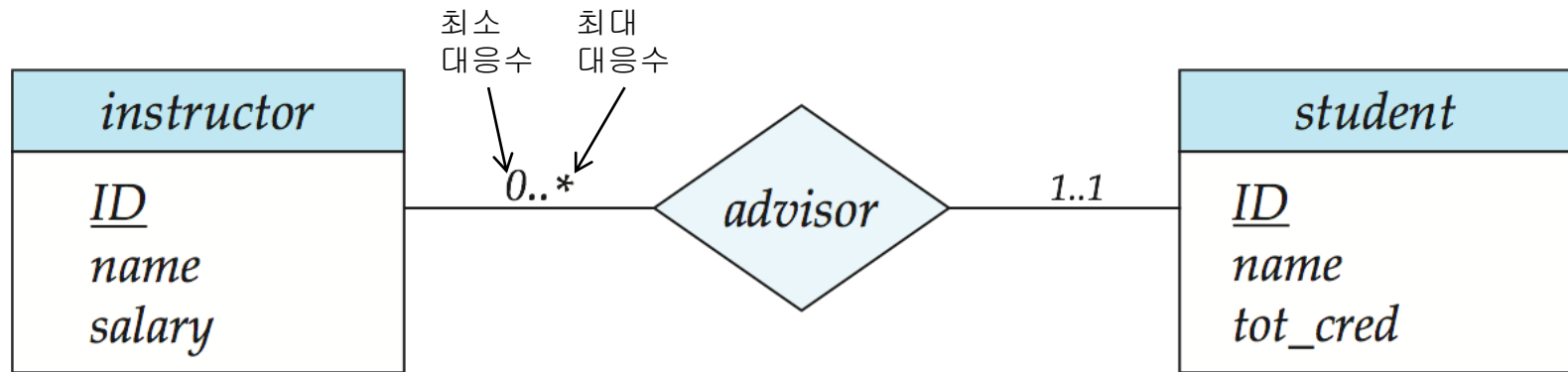
다대다 관계

- 한 명의 instructor는 advisor관계를 통해 여러 명의 student들과 관련
- 한 명의 student는 advisor관계를 통해 여러 명의 instructor들과 관련



대응수에 대한 대체 표기법

- 대응수 제한은 참여 제약조건을 표시함



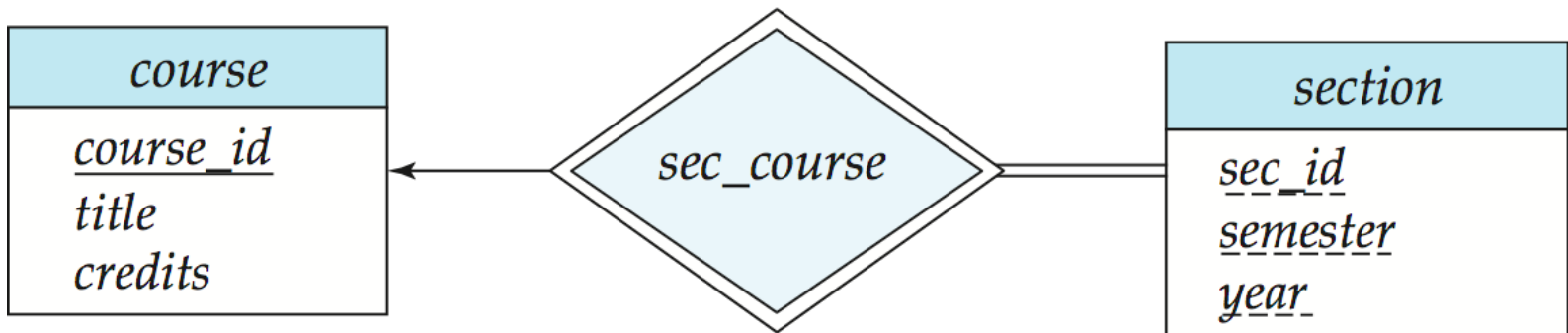
0..* → 각 교수님은 지도학생이 한 명도 없을 수도 있고, 여러 명 있을 수도 있다.

1..1 → 학생은 반드시 한 명의 지도교수님이 있어야 한다.

(학생 객체에 있는 모든 튜플들은 advisor 관계에 모두 참여 = “전체참여”)

관계 집합에 대한 개체 집합의 참여

- **전체 참여(total participation)** : 개체 집합에 있는 모든 개체들이 관계 집합에서 적어도 하나 이상의 관계에 참여하는 경우
 - 예제: 관계 집합 `sec_course`에 대한 개체 집합 `section`의 참여는 전체 참여 (과목이 없는 분반은 있을 수 없음)
- **부분 참여(partial participation)** : 어떤 개체들이 관계 집합에서 어떤 관계에도 참여하지 않을 수 있음
 - 예제: 관계 집합 `advisor`에 대한 개체 집합 `instructor`의 참여는 부분 참여

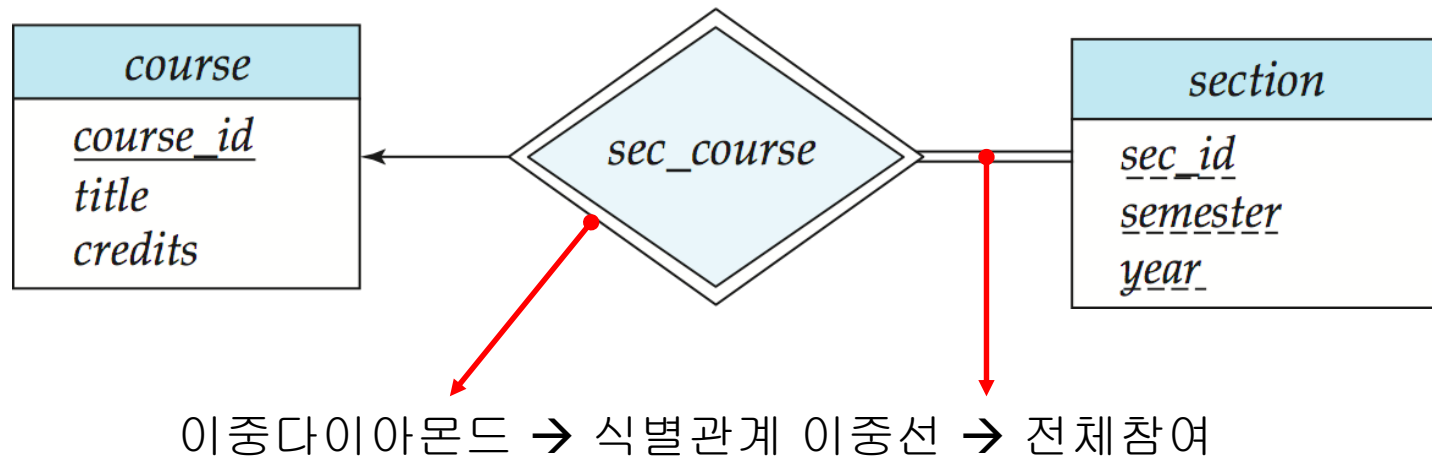


약성 개체 집합(**weak entity sets**)

- 주키를 가지고 있지 않은 개체 집합
- 주키를 가지고 있는 개체 집합 → **강성(strong) 개체 집합**
- 약성 개체 집합의 존재는 연결된 **식별 개체 집합(identifying entity set)**의 존재에 종속적임
 - 즉, 약성 개체 집합은 식별 개체 집합과 전체 참여, 일대다 관계 집합으로 연결되어야 함
 - **식별 관계(identifying relationship)**은 이중 다이아몬드로 표시
 - 약성 개체 집합에 존재하는 개체들을 식별하는데 사용되는 속성들의 집합 → **식별자(discriminator, 혹은 부분 키)**
- 약성 개체 집합의 주키
 - 연결된 강성 개체 집합의 주키 + 약성 개체 집합의 식별자
 - 예제:
 - *section*의 주키 = *course*의 주키(*course_id*) + *section*의 식별자(*sec_id, semester, year*)

약성 개체 집합 (Cont.)

- 약성 개체 집합의 식별자는 점선을 이용하여 밑줄로 표시
- 약성 개체의 식별 관계는 이중선으로 표시
- *section* 주키 – (*course_id*, *sec_id*, *semester*, *year*)



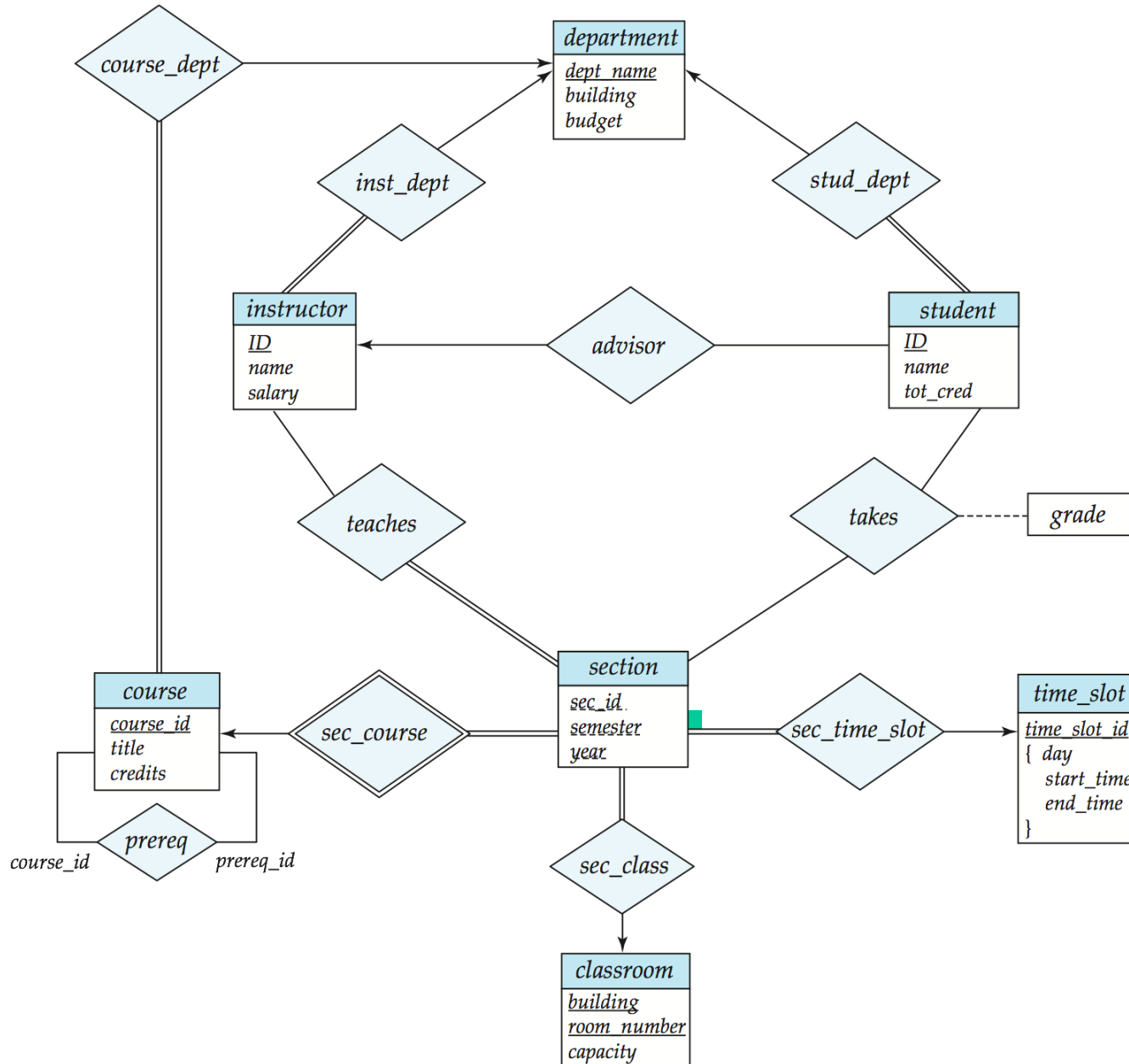
약성 개체 집합 (Cont.)

- 참고: 약성 개체 집합에 강성 개체 집합의 주키를 명시적으로 표시하지 않음 (식별 관계 집합에 묵시적으로 포함되기 때문)
- 만약, `course_id`를 명시적으로 표시하면 `section`은 강성 개체 집합이 됨. 그러면 `course`와 `section`에 존재하는 공통 속성 `course_id`는 관계 집합 `sec_course`에도 중복 존재하게 됨

주의1: E-R 다이어그램에서 `section`에서 `course_id`는 생략 가능하지만, 실제 DB 구현시 `course_id`가 있어야 한다.

주의2: `sec_id`를 `unique`하게 만들면 `section` 객체집합은 중복되지 않는 주키를 가질 수 있다. 그러나 개념적으로 `section`의 존재는 `course`에 의존적이기 때문에 바람직하지 않다.

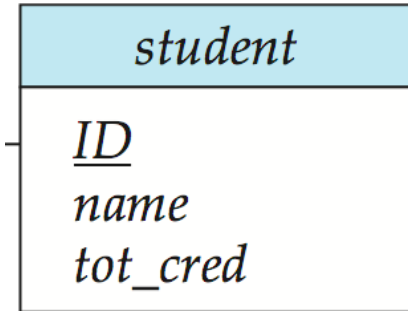
University를 위한 E-R 다이어그램



7.6 E-R 다이어그램을 릴레이션 스키마로 변환

- 개체 집합과 관계 집합은 데이터베이스의 릴레이션 스키마로 일관되게 표현할 수 있음
 - 단순 속성을 갖는 강성 개체 집합의 표현
 - 복합 속성을 갖는 강성 개체 집합의 표현
 - 약성 개체 집합의 표현
 - 관계 집합의 표현
- E-R 다이어그램은 데이터베이스를 구성하는 릴레이션 스키마들로 표현될 수 있음
- For each entity set and relationship set there is a unique schema that is assigned the name of the corresponding entity set or relationship set.
- Each schema has a number of columns (generally corresponding to attributes), which have unique names.

단순 속성을 갖는 강성 개체 집합의 표현



강성 개체 집합은 동일한 이름의 속성들을 갖는 스키마로 표현 가능

student(*ID*, *name*, *tot_cred*)

복합 속성을 갖는 강성 개체 집합의 표현

instructor

ID
name
 first_name
 middle_initial
 last_name
address
 street
 street_number
 street_name
 apt_number
 city
 state
 zip
 { *phone_number* }
 date_of_birth
 age ()

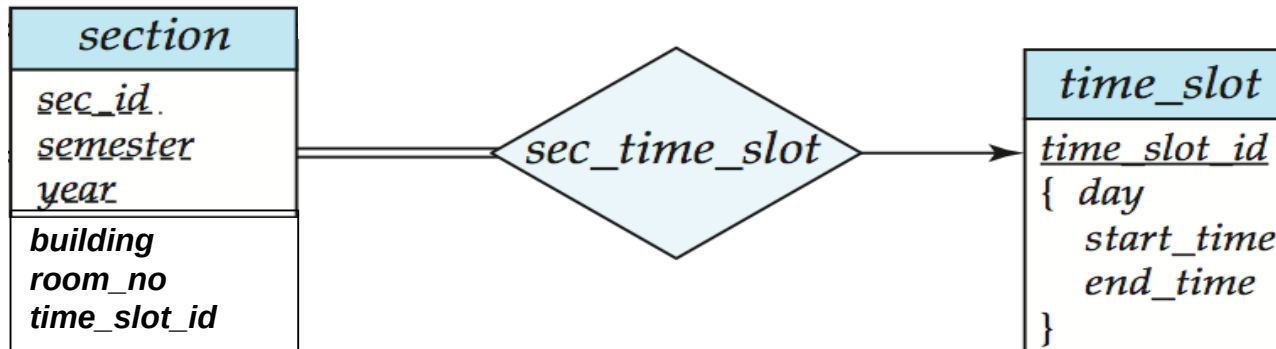
- 복합 속성은 각 요소 속성들을 분리된 개별 속성으로 flat하게 표현 가능 (복합 속성 이름을 prefix로 사용 가능)
 - 예/제: *name*속성을 구성하는 *first_name*, *middle_initial*, *last_name*는 *name_first_name*, *middle_initial*, *name_last_name*
 - 모호성이 없다면 prefix 생략 가능
- 다중-값 속성을 제외한 instructor 스키마
 - instructor*(ID, *first_name*, *middle_initial*, *last_name*, *street_number*, *street_name*, *apt_number*, *city*, *state*, *zip_code*, *date_of_birth*)

복합 속성을 갖는 강성 개체 집합의 표현(Cont.)

- 개체 E의 다중-값 속성 M은 분리된 스키마 EM으로 표현 가능
 - 스키마 EM은 E의 주키 속성과 다중-값 속성 M으로 구성
 - 예제: 다중-값 속성 *phone_number* 는 아래의 새로운 스키마로 표현 가능
 - inst_phone* = (ID, phone number)
 - 다중-값 속성의 각 속성값은 릴레이션 스키마 EM의 각 튜플에 대응됨
 - 예를 들어, 주키 22222 갖는 어떤 instructor가 phone number로 456-7890와 123-4567 번호를 가지고 있다면, 두 개의 튜플로 표현
 - (22222, 456-7890) and (22222, 123-4567)

복합 속성을 갖는 강성 개체 집합의 표현(Cont.)

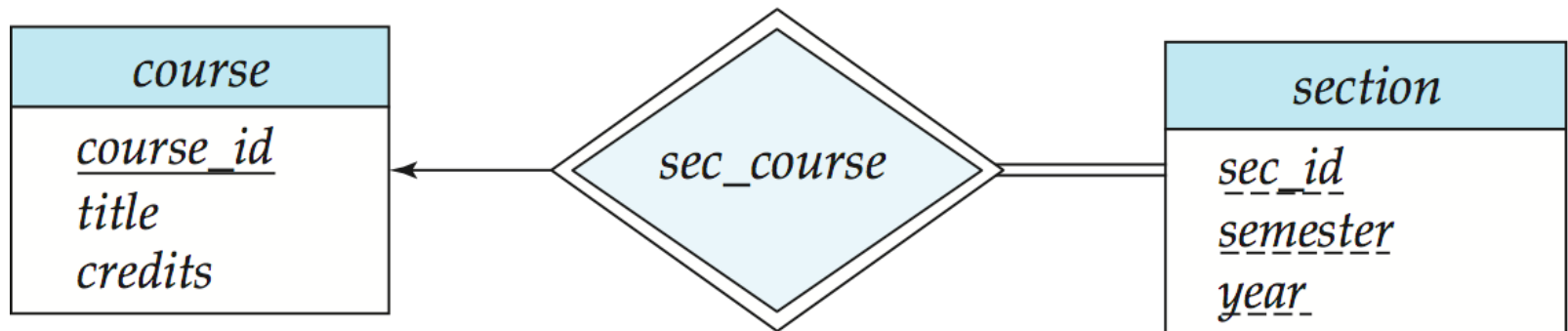
- 특별한 경우
 - 객체 *time_slot*은 주키 이외에 하나의 다중-값 속성만 갖는다.
 - 앞의 방법 이용시 같은 스키마를 갖는 릴레이션 생성
 - 해결 방안: 새로운 릴레이션을 생성하지 말고 복합 속성을 flat화하는 방법을 적용
 - *time_slot(time_slot_id, day, start_time, end_time)*
 - 주의사항: *section*의 속성 *time_slot_id*은 더 이상 *time_slot*을 참조하는데 사용되는 외래키가 될 수 없음



약성 개체 집합의 표현

약성 개체 집합은 연결된 강성 개체 집합의 주키를 자신의 속성으로 갖는 릴레이션으로 표현될 수 있음

section (course id, sec id, sem, year)



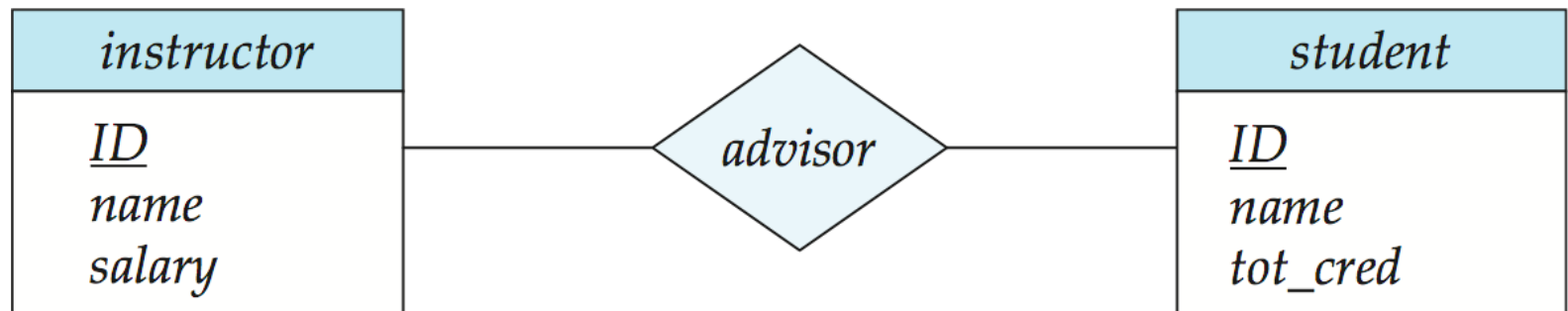
sec_course 스키마 = (*course_id*, *sec_id*, *sem*, *year*)

객체집합인 *section*의 스키마와 관계집합인 *sec_course*의 스키마가 동일하다.
→ 중복 제거 필요성

관계 집합의 표현

- 다대다 관계 집합은 관계에 참여하는 개체 집합들의 주키들로 구성된 릴레이션으로 표현
 - 추가적인 설명 속성을 갖을 수 있음
- 예제: 관계 집합 *advisor*의 스키마

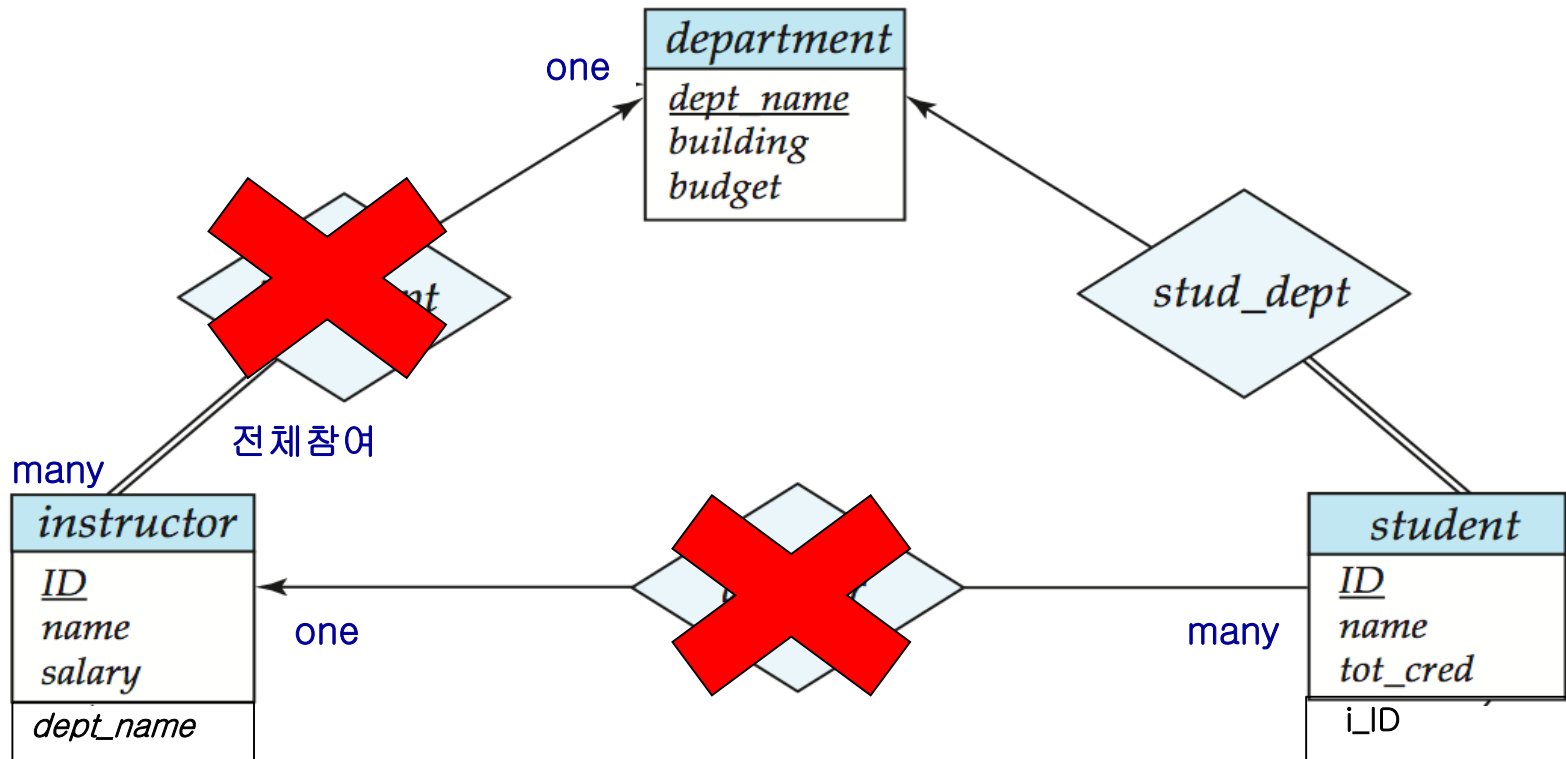
$advisor = (\underline{s_id}, \underline{i_id}) + (\text{설명속성, 예 date})$



스카마의 중복

다대일, 일대다 관계 집합(“다”측은 전체 참여)은 “일”측의 주키를 “다”측의 속성으로 포함시키고 생략 가능

예제: 관계 집합 *inst_dept*를 위한 새로운 릴레이션을 생성하지 말고 *instructor* 개체에 속성 *dept_name*를 추가함으로 해결 가능



스키마의 중복 (Cont.)

일대일 관계 집합

관계에 참여하는 양쪽 릴레이션 중 한쪽에 다른쪽 주키를 넣고, 관계를 삭제!

If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values

→ instructor(one)와 student(many) 에서 student가 부분 참여이면, 지도교수님이 없는 학생이 있을 수도 있다는 것을 의미하며 이 경우 student에 삽입된 instructor의 ID는 null이 된다

The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.

Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema

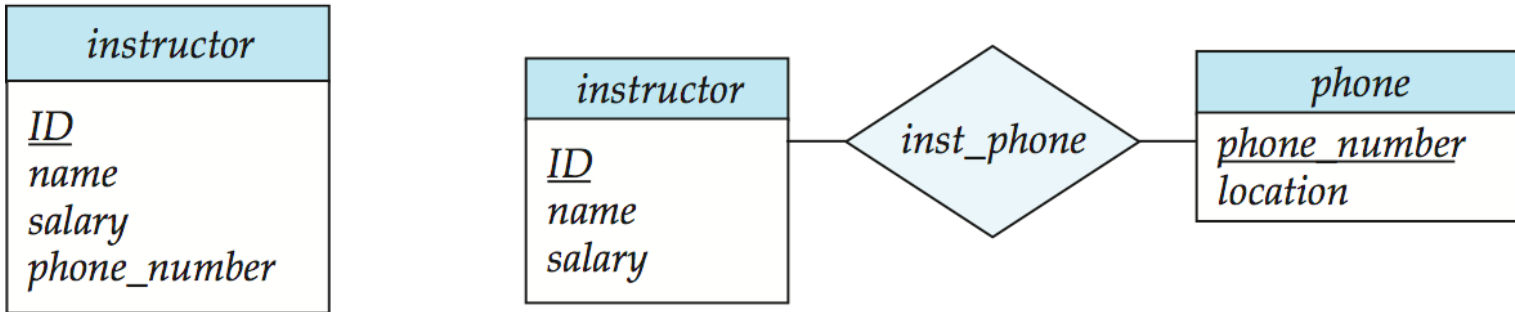
→ 앞의 예에서 course와 section을 연결하는 sec_course 스키마의 경우 이미 section schema와 동일

스키마의 중복 (Cont.)

- 일대일 관계 집합
 - 관계에 참여하는 양쪽 릴레이션 중 한쪽에 다른쪽 주키를 넣고, 관계를 삭제!
- 만약, “다”측의 참여가 부분 참여면, “다”측 스키마에 추가되는 속성 값이 널값이 될 수 있음
 - *instructor*(“일”)와 *student*(“다”) 에서 *student*가 부분 참여이면, 지도교수님이 없는 학생이 있을 수도 있다는 것을 의미하며 이 경우 *student*에 삽입된 *instructor*의 *ID*는 *null*이 됨
- 약성 개체 집합과 강성 개체 집합을 연결하는 관계 집합의 스키마는 중복됨
 - 예제: 앞의 예에서 *course*와 *section*을 연결하는 *sec_course* 스키마의 경우 이미 *section* 스키마와 동일

7.7 개체-관계 설계 이슈

- 개체 집합으로 표현할 것인가? 속성으로 표현할 것인가?

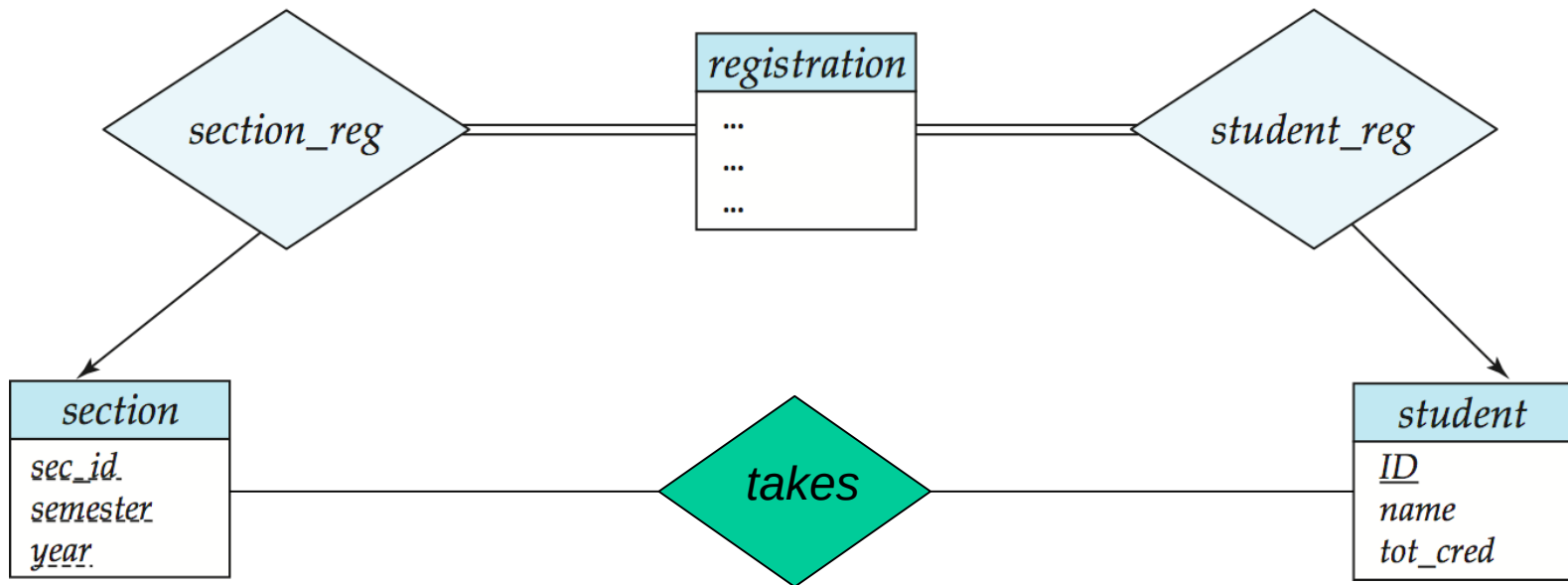


- phone 을 새로운 개체로 표현할 것인가? 아니면 속성으로 표시할 것인가?

개체-관계 설계 이슈 (Cont.)

- 개체 집합으로 표현할 것인가? 관계 집합으로 표현할 것인가?
 → 개체들 간의 발생하는 동작일 경우에는 관계 집합으로 표현하는 것이 바람직함!

수업등록기록 → 각 section에 대한 학생들의 등록현황을 파악하는데 더 많은 정보 제공 가능



takes ≠ registration ?

개체-관계 설계 이슈 (Cont.)

- 이진 관계 vs. n-진($n > 2$) 관계 집합
 - 비록 모든 n-진 관계는 2진 관계로 표현이 가능하지만 간혹 n-진 관계가 개념상으로 더 명확한 경우가 있음
 - 3진 관계
 - *parent* (father, mother, child)
 - *pro_guide* (project, student, instructor)
- 관계 집합의 속성
 - 관계 집합 *advisor*의 *data*라는 속성은 *student* 객체의 속성으로 배치해도 됨
 - 예: *instroctor-to-student*(일대다) 경우
 - *advisor*의 설명속성 *data*를 many측(*student*)의 속성으로 옮기는 것이 가능 → 일대일의 경우는 어느 쪽이든 가능

참고: n-진 관계를 2진 관계로 변환하는 방법

- In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.
 - Replace R between entity sets A , B and C by an entity set E , and three relationship sets:
 - R_A , relating E and A
 - R_B , relating E and B
 - R_C , relating E and C
 - Create a special identifying attribute for E
 - Add any attributes of R to E
 - For each relationship (a_i, b_i, c_i) in R , create
 - a new entity e_i in the entity set E
 - add (e_i, a_i) to R_A
 - add (e_i, b_i) to R_B
 - add (e_i, c_i) to R_C

